Peza Enio
Magnani Patrick

Optimal Algorithm

Experiment 1, Experimentation & Evaluation 2024

Abstract

This study examines the efficiency of three sorting algorithms—Selection Sort, Bubble Sort, and Quick Sort—by analyzing their execution times across varying array sizes, data types, and contents. The experiment was conducted to assist Bubble Inc. in selecting the most efficient algorithm for large datasets, with implications for other potential users facing similar needs. Controlled tests were run on machines with differing hardware specifications, and Java's `System.nanoTime()` measured runtime in nanoseconds. Results revealed that Quick Sort consistently outperformed Selection Sort and Bubble Sort, contrary to our hypothesis. This performance disparity highlights the influence of array size and type on sorting efficiency, suggesting that Quick Sort is optimal for diverse data inputs.

1. Introduction

The experiment's main purpose is to show which sorting algorithm is the most optimal choice for usage purposes regarding execution time. We are doing this research to aid the Bubble Inc. company in choosing the most efficient algorithm and in hopes also of aiding future possible users to help facilitate the selection of the fastest sorting algorithm.

Hypotheses:
Out of all the sorting algorithms, SelectionSort will yield the best time compared to BubbleSort and QuickSort in spite of the length and the type of the one dimensional array .

2. Method

2.1 Variables

Independent variable	Levels
Array Size	10,100,1000,5000,10000
Array Data Type	int,long,string,byte,char,float and double
Array Content	Numbers (random numbers from a set range) , Strings (from a file), Char (random values from within the ASCII interval converted to a char)

Dependent variable	Measurement Scale
Runtime	Nanoseconds

Control variable	Fixed Value
Reserved Memory Space (for each data type : int,long ecc)	Bytes (different from type to type but the value is constant for each type in the memory , never changes)
Device/OS	USI Laptop (Ubuntu 20.04/ 22.04) , Desktop (Windows 11)

2.2 Design

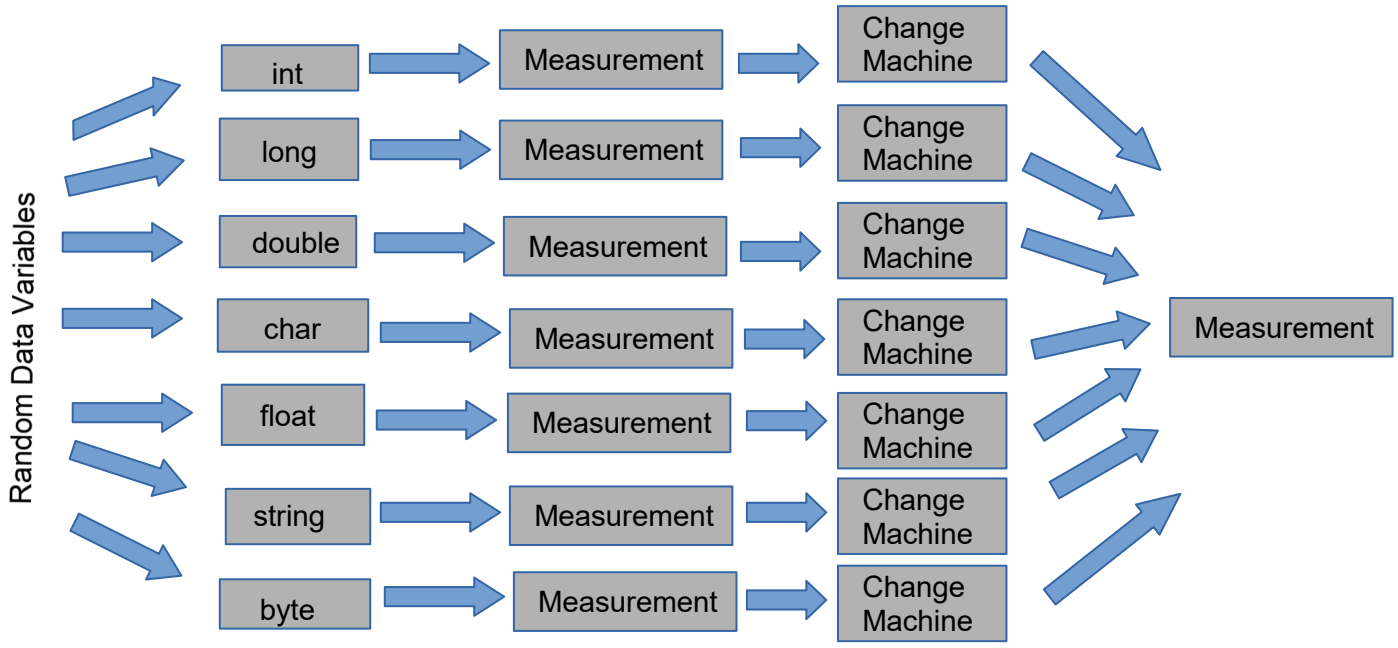
Type of Study (check one):

☐ **Observational Study**	☐ **Quasi-Experiment**	X **Experiment**

Number of Factors (check one):

☐ **Single-Factor Design**	X **Multi-Factor Design**	☐ Other

The study that is being conducted is classified as an experiment because of the direct manipulation of different variables such as array size ,data type and array contents. The variables are under control , allowing systematically to test their impact on the different sorting algorithms times . The experiment has a Multi-Factor Design because it involves different independent variables . These variables can influence the processing time independently or in combinations with one another.



2.3 Apparatus and Materials

The experiment was conducted on 2 different devices : a laptop and a Desktop ,
The laptop contains : CPU → i7-12700H , RAM → 16 GB DDR4 , HDD
Laptop Model :→ Precision Dell 5570 ,
The Desktop contains : CPU → Ryzen 9 7900x3D , RAM → 32 GB DDR5 6000 MHz , SSD

The laptop runs Ubuntu , one runs Ubuntu 20.04 , the other runs Ubuntu 22.04 . As for the Desktop it runs a Windows 11 copy .

The software used to run the experiment is IntelliJ IDEA Ultimate 2024 using Java 21 and the runtime was measured using System.nanotime() .

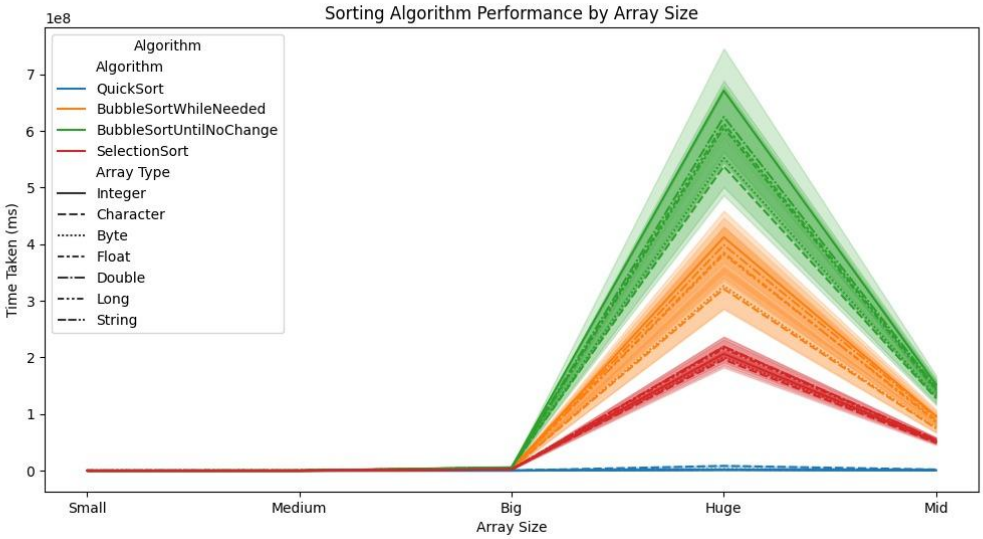
2.4 Procedure

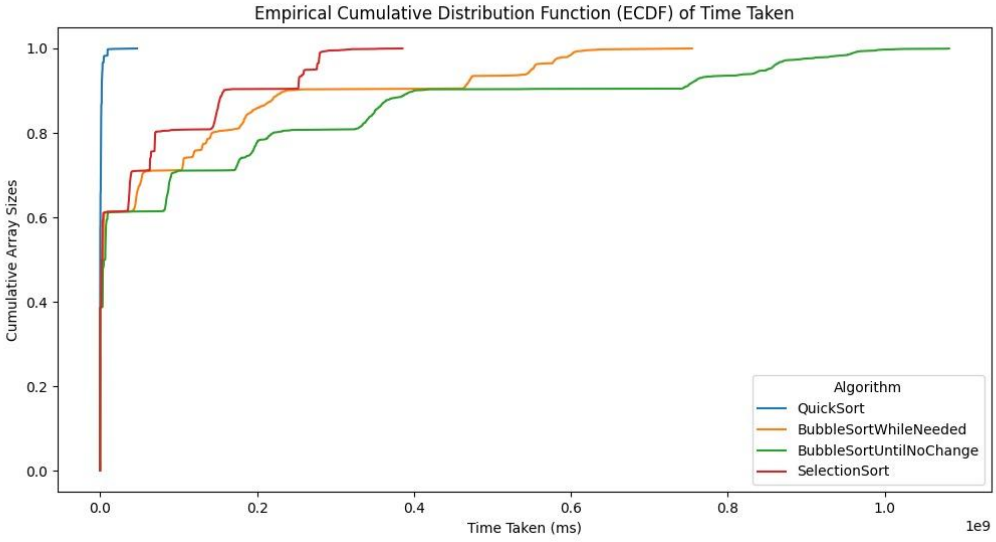
The experiment was conducted on 3 different machines . Each in a different state : one machine was under stress for two hours , one was freshly boot up and the last machine had been running for 4 hours . Each machine ran the test in all the states mentioned above . The results don't change that much with regards to the states of the machine . This might be due to the fact that the time frame for which the machines were put under stress is small .

The code was run thirty times , and the data was saved and processed at the end of the experiment using a Python script . The order in which the code was run on the machines is irrelevant to the final outcome of the experiment .

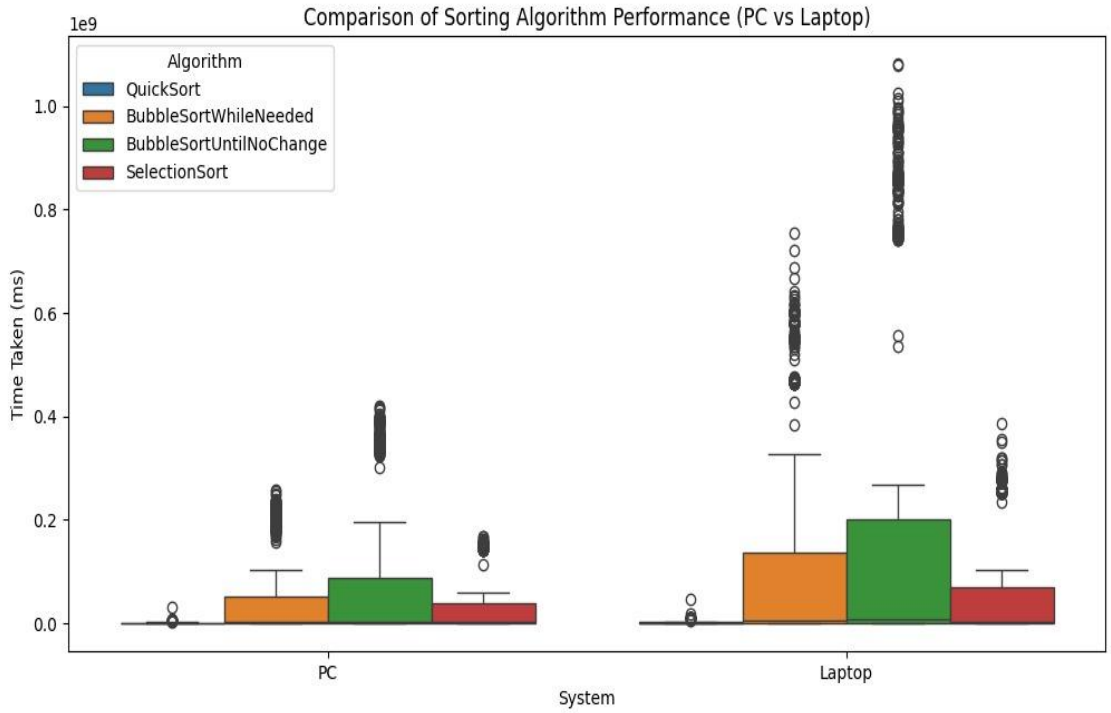
3. Results

3.1 Visual Overview





3.2 Descriptive Statistics



The results clearly show that the minimum is near 0 ms and the first quartile is around the 0.04 ms . This means that the average minimum time is around 0.02 ms. The median is clearly shown as 0.1 ms . The third and maximum value are quite high compared to the average due to an exponential size increase of the array . The third quartile is leaning towards 0.2 with a value of approximately 0.18 ms . The maximum value is situated at over 1ms due to the inefficiency of the sorting algorithm and the slower machine.

4. Discussion

4.1 Compare Hypothesis to Results

The results that were obtained prove that the hypothesis chosen was wrong . The hypothesis sustained that Selection-Sort would have been the fastest out of all the algorithms in spite of size and data type . This was not the case as clearly shown in the two

graphs (3.1) where Quick-Sort was proved to be the faster one in spite of all the conditions mentioned above .

The hypothesis was wrong possibly due to the worst case scenario for both algorithms and the size of the array . It is clearly shown In the pictures that at a smaller size of the array ,the time difference is almost nonexistent .While a computer can see these differences , human beings can not comprehend such small differences which led to the wrong hypothesis .

4.2 Limitations and Threats to Validity

The study was limited by the current advancements in software development and machine power . If the experiment were to be run with a much bigger array size , both the software and the machine would have not been able to complete the task . This can be overcame only by future developments in the software and computer manufacture .

4.3 Conclusions

The main conclusion that can be drawn from the study is that the fastest algorithm is the one with the best – worst case scenario that doesn't have to iterate the array multiple times , in this case Quick-Sort . It is also shown that the power of the machine does not affect and/or change this fact but it can aid in running said algorithm within a shorter time span .
Also , the OS does not affect the final outcome.

Appendix

A. Materials

The resources that contributed to the development of this study include :

- Book from the course
- Slides presented during class
- Online documentation for both Java and Python : - https://matplotlib.org/stable/index.html
  - https://docs.oracle.com/en/java/javase/22/

B. Reproduction Package (or: Raw Data)

All the code and documentation can be found on the following Git Hub repository :
https://github.com/Malty44/Experimentation_P1.git